1:24000+ Scale Hydro Clearinghouse
Client transaction process

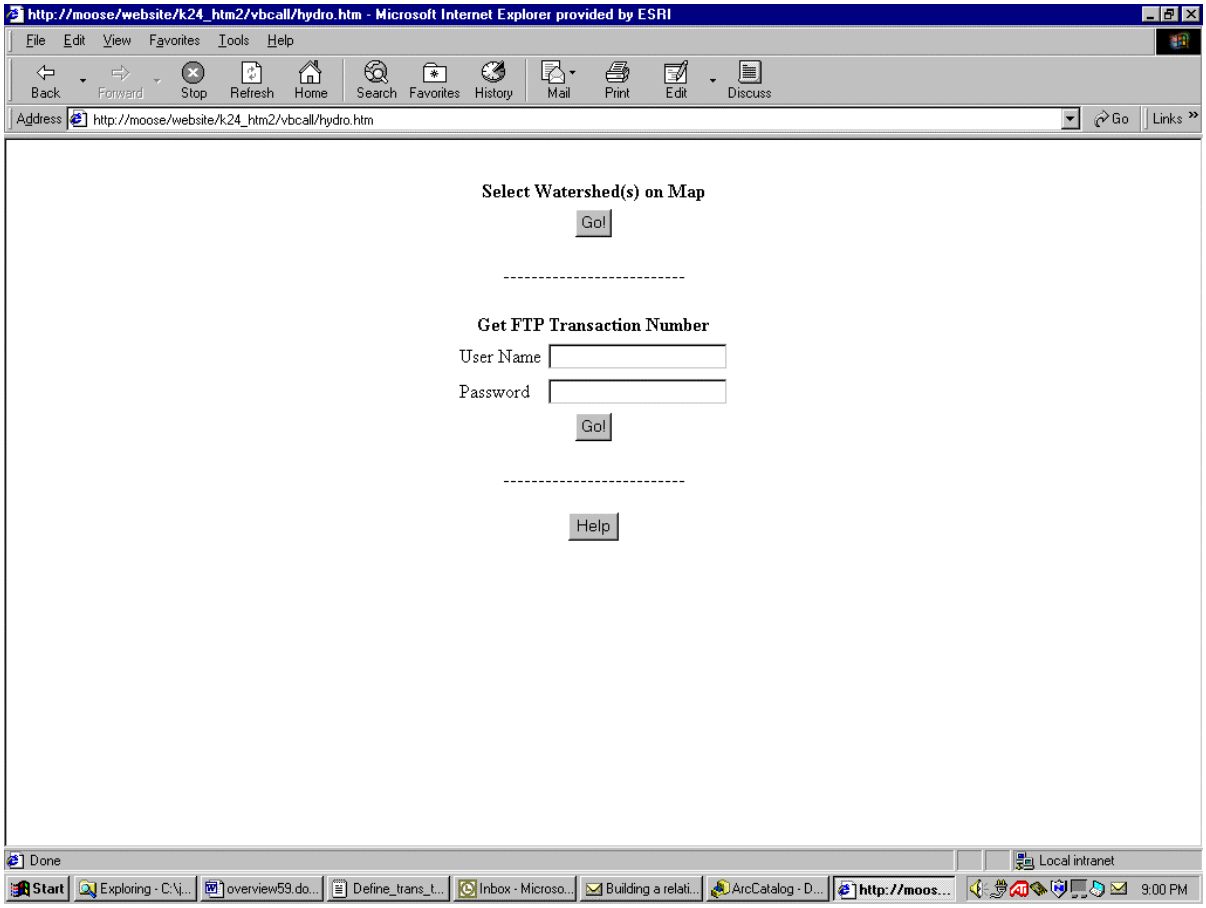
The process flow for data checkout, editing, and checkin is described by tracking a typical transaction. This process involves actions by the client, and responses by the server. The overall transaction process consists of the following major steps:

- User identifies self and starts a transaction
- User submits a checkout polygon covering desired area of work
- Server locks the area
- Server verifies checkout and extracts data to edit
- User copies data from server and edits the data
- User tests data
- User submits data
- Server recieves and tests data
- Clearinghouse Manager accepts submission and posts changes, or rejects submission, in anticipation of a re-submission by the user, or aborts whole transaction.
- Server releases the lock on the area

These basic steps are detailed in the following sample transaction flow between a clearinghouse client and the clearinghouse server. Some of the steps are performed using a web browser as the client, and some are performed using a client-side ArcInfo AML application supplied as part of this contract.

User identifies self and starts a transaction

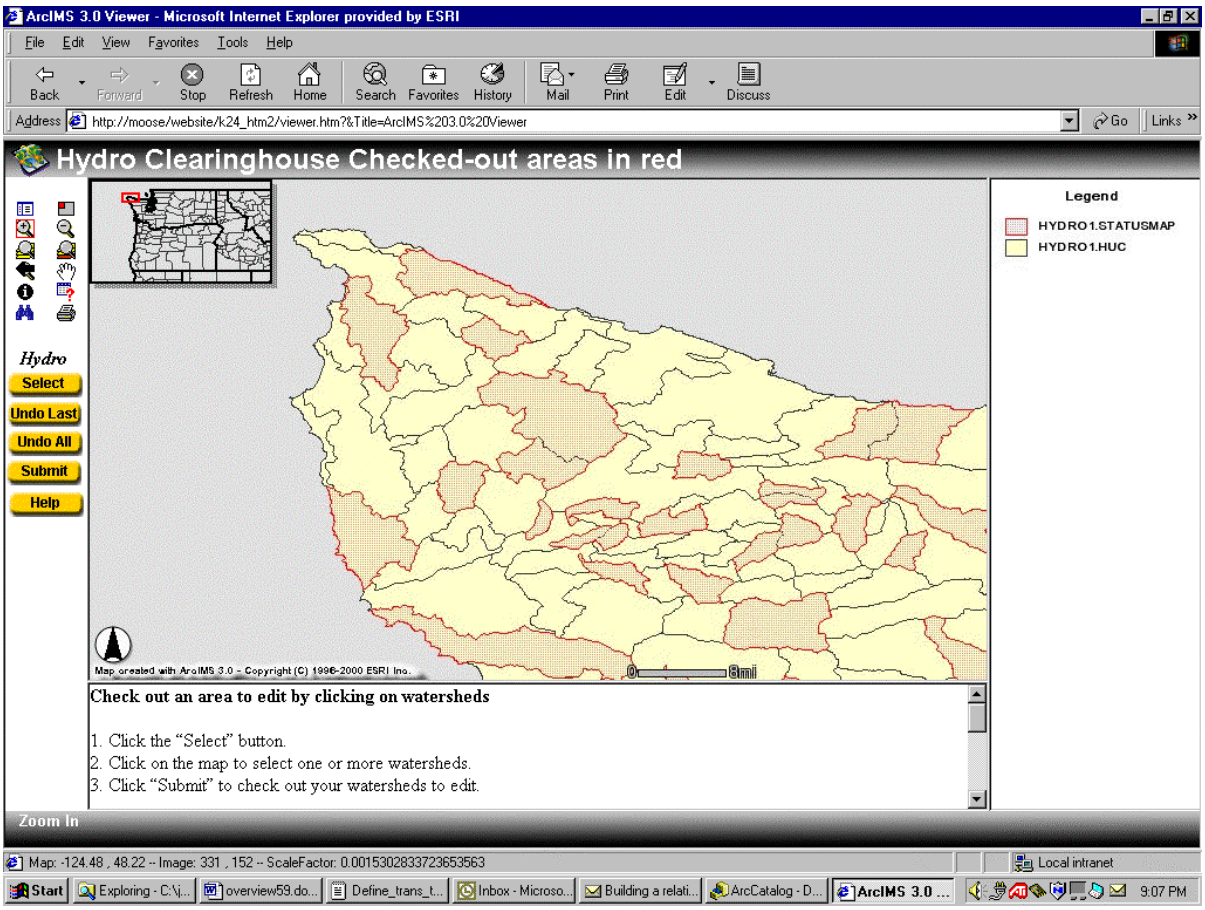
- The user connects a Java-enabled web browser (Internet Explorer or Netscape) to the clearinghouse web site.



- The user selects “Select Watershed(s) on Map” or “Get FTP Transaction Number”

If the user selected “Select Watershed(s) on Map”:

- The following page is displayed:



- The user hits the "Select" button, then clicks on one watershed that is not red. Red indicates a checked-out area.
- If the user selects a second point, a line is drawn.
- If the user selects three or more points, a polygon is drawn.
- All watersheds under the point, line, or polygon are selected.
- The user hits the "submit" button
- A form pops up requiring the user to enter a username and password.
- The user is notified on the screen and by email of the transaction id.
- The watersheds are merged into a single polygon. This polygon is written to a coverage, named c<transaction_id> (example: transaction 356's checkout polygon coverage would be named c356, and its export file would be named c356.e00). This coverage is exported, and placed in the hydro_ftp\sc24000\checkout_covs directory.
- An "end flag" file is then placed in the hydro_ftp\sc24000\checkout_covs directory. This is a signal file signifying that the file transfer of the main file is complete. The end flag file is named e<transaction_id>.txt (e.g. e356.txt)
- This file will be processed by the Clearinghouse Server to check out data, as described in the step after next.

If the user selected "Get FTP Transaction Number":

- The user enters a username and password, and clicks "GO!".

- The user will use ArcInfo offline to create a polygon coverage containing one polygon defining the area the user wants to edit. This coverage must be clean, without label errors, containing one real polygon and the universe polygon. It must be double precision, and stored in decimal degrees, NAD83. It must have a valid PRJ file. Any attributes will be ignored. Then the user will export and FTP this polygon coverage to the server.
- The name of the coverage must be c<transaction_id> and it's export file must be named c<transaction_id>.e00.
- The user is instructed to then FTP an "end flag" file to the hydro_ftp\sc24000\checkout_covs directory. This is an empty file which signifies that the file transfer of the main file is complete. The end flag file is named e<transaction_id>.txt (e.g. e356.txt)
- An AML named FTP_CHECKOUT_POLY.AML is provided to automate this process.

Server verifies checkout and extracts data to edit

- In either case, an export file and an "end flag" file show up in the hydro_ftp\sc24000\incoming\checkout_polys directory. This is the queue of pending checkouts. The server periodically scans this directory, and looks for files that have finished uploading, as signified by the presence of an "end flag" file.
- A directory, named t<transaction_id> (e.g. t356), is created in the hydro_server\sc24000\transactions directory. Only the server has access to this directory.
- This event is logged in the transaction table.
- The export file is imported to a coverage named edit_boundary and validated:
  - One contiguous polygon
  - The export file name is the same name as transaction to which it applies
  - The server compares the polygon to the checkout polygon layer.
  - If there is no overlap with active transaction polygons, then this polygon is added to the STATUSMAP layer, with an attribute identifying its transaction_id
- If it fails, mail is sent to the client, explaining the problem. The transaction is aborted. This event is logged in TRANSACTION_HISTORY as CKOUT_POLY_REJECT, and the transaction is closed with a disposition of CKOUT_POLY_REJECT in the TRANSACTION table.
  - The user must now go back to the web site and start a totally new transaction
  - The transaction is closed
- If the checkout succeeded, the following sequence of events occurs:
  - The checkout polygon is inserted into the statusmap layer
  - In a future version, a new statusmap coverage is extracted to the hydro_server\sc24000\statusmap directory, showing the areas for active transactions only. These polygons are non-overlapping. This coverage is exported and placed in the hydro_server\sc24000\statusmap directory. The readme.txt file is placed in the hydro_server\sc24000\statusmap directory, indicating that the new

statusmap.e00 is ready for download. These readme.txt files are used in several places in this system. A readme.txt file in this system serves two purposes:

- It documents the purpose of the data in its directory, as well as contextual information such as date/time, transaction id, and username.
- Its presence serves as a signal that the other data in the directory is not in the process of being copied, because the readme.txt file is always copied into the directory last.

- The directory hydro_server\sc24000\transactions\t<transaction_id>\checked_out is populated with checked-out data for this transaction. The following coverages and tables are extracted from the SDE database and stored here:
  - wc – watercourse routes are extracted if they fall completely within the checkout polygon. Event table rows are extracted if the LLID matches the extracted stream route.
  - wb - waterbody regions are extracted if they fall completely within the checkout polygon.
  - ws - waterbody shoreline routes are extracted if they fall completely within the checkout polygon. Event table rows are extracted if their LLIDs match the extracted waterbody shoreline routes' LLIDs.
  - Note: Event tables have same root name as the coverage, so they copy with the coverage.
  - wp - Water Points
  - Rows from the water point relate table are extracted if their LLIDs match extracted water points
  - Now we have four coverages of extracted data, and one coverage containing the checkout polygon.
  - Now waterbodies, shorelines, and watercourses that touch but are not completely within the checkout polygon are exported as background coverages named wb_back, ws_back, and wc_back, respectively. No background coverage is needed for water points, since a point has zero dimensions and physically cannot fall partially within the checkout polygon.
  - When the user sends back edits, we need all 4 export files back, even if changes were not made. The contents of these coverages will totally replace the extracted features, regardless of how few, if any, edits were made.
- LLID tables are populated with ids of all features and event/relate rows checked out.
- hydroqa.aml (quality assurance) is run against the checked-out data. If errors are detected, the transaction proceeds anyway. Data will be checked on insert, so errors in the database are not anticipated. However, if the database is populated using a batch load, or if the clearinghouse manager chooses to load data with known errors, or if additional QA/QC tests are implemented in the future, it is conceivable that there will be data in the server that does not pass the QA/QC tests. This step helps the prospective editor be aware of problems that may be encountered.

- The error report is generated as a file named hydro_server\sc24000\transactions\t<transaction_id>\checked_out\checkout_errors.txt
- This error report file is copied to hydro_ftp\sc24000\outgoing\checkout_covs\t<transaction_id>\checkout_errors.txt for the user to download.
- This event logged to transactions table
- The eight coverages that were extracted are exported directly to the hydro_ftp\sc24000\outgoing\checkout_covs\t<transaction_id> directory. Once the last coverage has been exported to this directory, a readme.txt file is placed there, to signal the user that the files are ready to download.
- Mail sent to client that the checked-out data is ready to download
- CHECKOUT_DATA_POST is logged to the TRANSACTION_HISTORY table

User copies data from server and edits the data

- The user gets email that the data for their transaction is ready to download.
  - The user runs hydrocheckout.aml, and enters the transaction id (e.g. t356) and FTP password. This AML does these things:
    - Creates a subdirectory named the transaction id (e.g.t356)
    - FTPs the eight .e00 files into this directory
    - Imports the .e00 files
    - Leaves the .e00 files in place in case the coverages are inadvertently corrupted by the user
    - FTPs the checkout_errors.txt file into this directory, which is an error report on the checked-out data. This error report was generated by the server when the data was extracted. The user can create a similar report by running hydroqa on the checked-out data. It is provided as a reference to compare against the error report that will be generated by hydroqa.aml immediately prior to checkin.
  - The user edits the wc, wp, ws, and wb coverages, using the edit_boundary, wb_back, ws_back, and wc_back coverages as background coverages.
    - The user can employ any tools at their disposal to edit these coverages. Examples are SRT and STEVE. These tools are being modified to comply with data model changes resulting from the integration of data from Oregon framework partners, Washington framework partners, and Federal partners.
    - The coverages can be edited in place, or copied for editing in another workspace. The only requirement is that when the edits are ready to be checked in, the final changed coverages must copied back to this workspace under the original names of wc_checkin, wp_checkin, ws_checkin, and wb_checkin. This directory is the staging area for final QA/QC and checkin.
    - All edits must fall within the edit_boundary coverage
    - The user should not modify the edit_boundary, wc_back, ws_back and wb_back coverages.

- The user is responsible for making sure the features line up properly between the wc, wp, and wb coverages
- The user is responsible for ensuring that features snap properly to the background coverages.
- The user is also responsible for ensuring that wb, wp, ws, and wc features line up properly with each other.
- LLID is an id value that is an encoded long/lat value for a point on each feature. The decoded long/lat for all LLID values created for new features must fall within the edit_boundary polygon. This ensures that two editors do not simultaneously create the same LLID for different features.
- LLID values do not need to be unique between the wc, wb, ws, and wp coverages. LLIDs must be unique within each coverage.
- Other QA rules are documented below. The above rules are listed here to clarify the roles of the edit_boundary, wc_back, ws_back, and wb_back coverages.

User tests data

- The client runs hydroqa.aml on the edited data, and corrects any errors detected.
- The reason for this step is to avoid frustration and wasted time from multiple submissions and rejections. The user can test the data locally instead. Since the data will be clean before it is sent to the clearinghouse, rejected submissions should not be common.

User submits data

- Now client FTPs edits back to the server by running hydrocheckin.aml. This AML exports the four edit coverages only. The other coverages are not supposed to be edited anyway, so there is no point in sending them back to the server.
  - An error report is run immediately prior to FTPing the export files to the server. Depending on the severity of the errors, and whether these errors are problems which were inherited when the data was checked out in the first place, the clearinghouse manager may or may not decide to allow this data back into the clearinghouse.
  - The error report is written to a file called checkin_errors.txt. If the checkin is aborted and retried later, the error report will be overwritten by the next checkin. There will be two error reports in the directory:
    - Checkout_errors.txt – downloaded error report on checked-out data
    - Checkin_errors.txt – locally generated error report on data for last attempted checkin.
    - Note: the user can run hydroqa.aml anytime they want, and name the error files anything they want. These two files are just the ones that automatically show up when the user checks out or checks in data. Their purpose is to report the error status of the data that was checked out and checked back in.

Server receives and tests data

- The server is constantly polling the hydro_ftp\sc24000\incoming\checkin_covs directory. When it encounters a subdirectory called t<transaction_id> (e.g. t356), it checks in that directory for a file named readme.txt. This is a signal file whose presence indicates that the client has finished populating this directory.
- The server now checks the transaction table and that the transaction_id refers to an open transaction that is ready for data submission. If not, then:
  - If any data shows up in the checkout_polys or checkin_covs directories that refers to prevoiusly active, currently dead transaction ids, (e.g. if the transaction id can be found in the transaction table), it is archived in hydro_server\sc24000\transactions\t<trans_id>\rejected\<date_time>, and mail is sent to the owner of the dead transaction.
  - If any data shows up in the checkout_polys or checkin_covs directories that refers to nonexistent transaction ids, (e.g. if the transaction id *cannot* be found in the transaction table), it is stored in hydro_ftp\sc24000\wastebasket, and mail is sent to ther clearinghouse manager. This directory can be cleaned up by the clearinghouse manager as desired.
- Files older than a month may be deleted from the wastebasket directory. The only reason the files are not deleted outright is to provide a mechanism for recovering from an error in transaction management that resulted in the submitted data being thrown away.
- If the transaction is valid:
  - the server moves the export files to a <date_time> directory in the hydro_server\sc24000\transactions\t<transaction_id>\submitted directory, then it deletes the hydro_ftp\sc24000\incoming\checkin_covs\ t<transaction_id> directory.
  - A CKIN_DATA_RECV code is added to the transaction_history table.
  - Mail is sent to the client confirming receipt of the data.
  - The server runs hydroqa.aml, and posts the report in hydro_server\sc24000\transactions\t<transaction_id>\submitted\<date_time>checkin_errors.txt
  - Mail is sent to the clearinghouse manager, informing him that data has been submitted.
  - A CKIN_PROCESSED code is added to the transaction_history table.

Clearinghouse Manager accepts submission and posts changes, or rejects submission, in anticipation of a re-submission by the user, or aborts whole transaction.

- The clearinghouse manager reviews the checkin_errors.txt file, displays the data on the screen for a quick check, and decides whether to accept or reject the submission. Comments are added to the checkin_errors.txt file, if desired.
- If the submission is rejected:
  - Mail is automatically sent to the user explaining why the submission was rejected.

    - The data is moved to hydro_server\sc24000\transactions\t<transaction_id>\rejected\<date_time>
    - A CKIN_REJECT code is added to the transaction_history table
    - The clearinghouse manager should generally contact the user directly, and explain the problem.
- If the submission is accepted:
    - Mail is automatically sent to the user verifying that the submission was accepted.
    - A CKIN_ACCEPT code is added to the transaction_history table
    - An SDE version named <transaction_id> is created. Database changes are made to this version.
    - The SDE logfile created at checkout time is used to delete all features or rows that were checked out.
    - The checked-in features and rows are inserted into this version.
    - This version is reconciled against the default version. No conflicts should occur. If so, urgent error mail is sent to the clearinghouse manager.
    - This version is posted to the default version
    - The checkout polygon is marked and dated as retired from the statusmap layer. This area is now ready for new edits.
    - A CKIN_ACCEPT code is added to the transaction_history table